ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOCPL01

IBEX: BEAMLINE CONTROL AT ISIS PULSED NEUTRON AND MUON SOURCE

K. V. L. Baker, F. A. Akeroyd, D. Keymer, T. Löhnert, C. Moreton-Smith, D. Oram
ISIS Neutron and Muon Source, Didcot, UK
J. R. Holt, T. A. Willemsen, K. Woods, Tessella, Abingdon, UK

Abstract

For most of its over 30 years of operation the ISIS Neutron and Muon Source [1] has been using bespoke control software on its beamlines. In the last few years, we have been converting the beamline control software to IBEX [2], which is based on the Open Source EPICS toolkit [3]. More than half the instruments at ISIS are now converted. IBEX must be robust and flexible enough to allow instrument scientists to perform the many experiments that they can conceive of. Using EPICS as a base, we have built Python services and scripting support and are developing an Eclipse/RCP Graphical User Interface (GUI) based on Control System Studio [4]. We use an Agile based development methodology with heavy use of automated testing and device emulators. As we move to the final implementation stage, we are handling new instrument challenges (such as reflectometry) and providing new functionality (live neutron data view, script generator and server). This presentation will cover an overview of the IBEX architecture, our development practices, what is currently in progress, and our future plans.

INTRODUCTION

The ISIS Neutron and Muon Source is located at the Science and Technology Facility Council (STFC) Rutherford Appleton Laboratory in Oxfordshire, UK. The facility produces beams of both neutrons and muons to conduct experiments at this world leading centre for research in the physical and life sciences. To successfully deliver these experiments the beamlines have to be controllable, and this is the function that IBEX provides.

There are more than 30 beamlines at ISIS covering 7 main science techniques. The facility was opened in 1985 and 10 years ago the second target station at ISIS was completed, doubling the capacity for beamlines. With that length of history, and the established reputation of ISIS, changes to control systems have to be done carefully.

IBEX ARCHITECTURE

Hardware

A detailed description of our hardware architecture has been presented in the past [5]. The core instrument control computers are a Virtual Machine (VM) which hosts the Experimental Physics and Industrial Control System (EPICS) Input Output Controllers (IOCs) for the beamline devices. These VMs are run one per beamline on a dedicated server, and are accessible on the general ISIS network. They will also have a dedicated private network where this is required for the devices being controlled. An EPICS gateway is used to provide access control to IOCS.

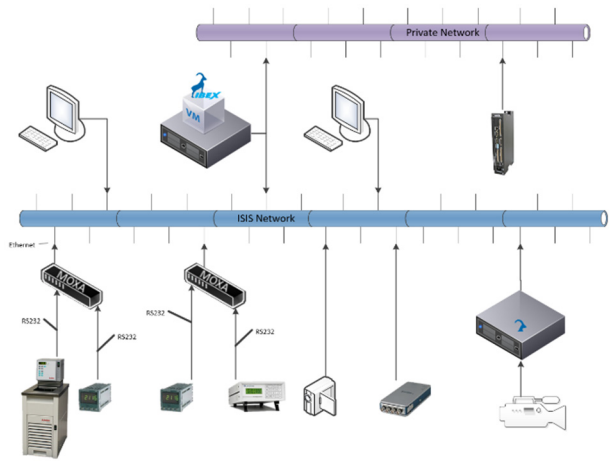

Figure 1: Hardware architecture.

More complicated devices typically have their own control computer, and will run a reduced version of IBEX to host the IOC more locally. An example of this are some of the imaging cameras used by the IMAT instrument. See Fig. 1 for an overview of the hardware architecture.

As the instrument control computer is a VM, there is usually an additional local computer provided to access the control system over the network by the user. IBEX's client-server architecture also enables users to view read-only information on remote computers without impacting local instrument control, which was not possible under the previous control system.

Software Architecture

A basic overview of the software architecture is shown in Fig. 2, and a more detailed description can be found in [2].

The IBEX Server controls individual devices via EPICS IOCs, which are wrapped within procServ [6] process harness instances. Each IOC has an associated config.xml file which provides information on configuration options (EPICS Macro names) for using within a beamline. The actual values for these macros (which might be used to set a communication port address) are maintained by the BlockServer process, which collates sets of IOC macros into "configurations" for selection by the user.

We run two Control System Studio [4] archive engines to store various Process Variables (PVs) of interest, one deals with scientifically useful data (which is placed into

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOCPL01

the experiment data files) and the other contains diagnostic data.

The neutron and muon data is generally captured via facility specific detectors and data acquisition electronics. This data is collected from the hardware and combined with PV data by the Instrument Control Program (ICP) to generate a NeXus [7] file which is then made available to the experimentalists for analysis of the data.

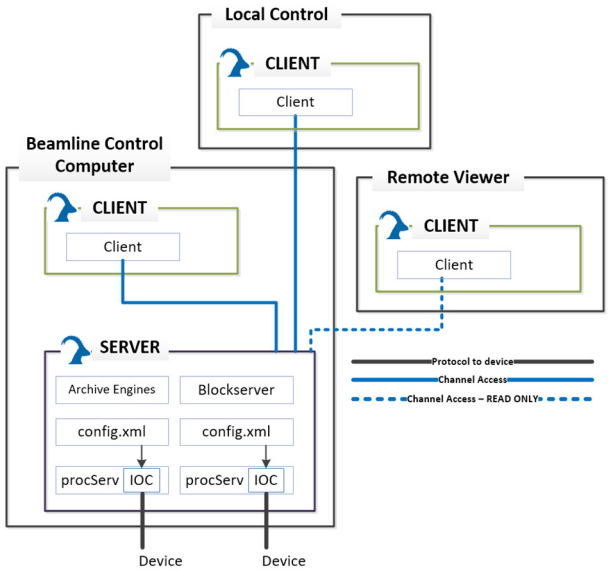

Figure 2: Software architecture.

The Blockserver is the keystone in IBEX, as it manages the configuration of the instrument and provides the flexibility the instrument scientists require. It is responsible for controlling which IOCs are running, and which PVs should be logged to the data files, based on an instrument configuration which is defined by the scientists. The Blockserver is a Python process based on PCASpy [8], which enables it to serve and receive values from clients via EPICS Channel Access.

As already stated the instrument configuration contains the list of IOCs to be run on that particular beamline or instrument, as well as the associated parameters, such as connection addresses. That list varies from beamline to beamline, and between experiments on a beamline, which is why the system needs to be as flexible as it is.

An instrument configuration can be built up from a number of smaller parts called components, this helps reduce duplication as fixed items on a beamline can be placed in a single component and included in all configurations easily.

Whilst all communication to and from the IBEX Server is via Channel Access, this isn't the easiest way to consider the system for an instrument scientist, as such a GUI known as the IBEX Client (see Fig. 3) has also been developed.

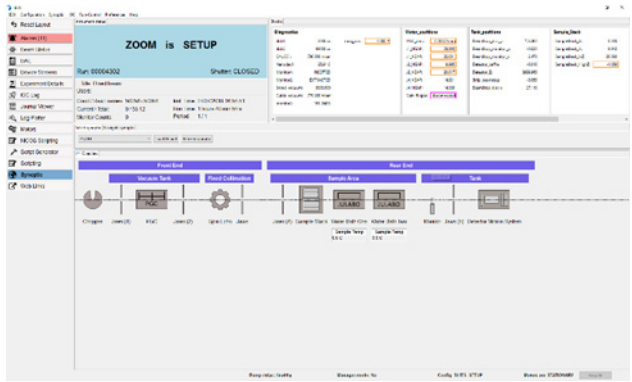

Figure 3: IBEX client.

The IBEX client is an Eclipse RCP based program, with many tabs able to show different information. The Scripting option provides a Python scripting window running the ISIS genie_python package [9]. This package provides a familiar command set to instrument scientists following from the Open GENIE [10] scripting language that was used with the previous control system.

The Synoptic window, as showing in Fig. 3, is an addition to the views available to instrument scientists. The icons shown are the devices along the beamline from the target on the left to the detectors on the right. Each icon can be clicked on to gain access to a "Device Screen" which is a CS Studio Operator Interface (OPI). These screens for equipment can also be independently added to the Device Screens window, allowing different ways of interacting with the equipment.

Due to the client/server nature of the IBEX system, the IBEX Client can actually be directed to get information from any IBEX Server on the network. This allows technicians to check on devices on multiple beamlines easily from their office computer systems.

THE TEAM

As is usually the case for software teams supporting science research the team working on developing IBEX are also the team providing support for it, dealing with immediate issues as well as finding longer term fixes to problems. The team is responsible for the computers used to control the beamlines, as well as the software needed at each level. Whilst the main infrastructure of the network is not our responsibility, we have to be aware of it and use it, and the ways to connect devices to the instrument control computers.

This makes the IBEX team a DevOps [11] team needing skills and knowledge ranging from the building blocks of the actual computer to being aware of accessibility for the Graphical User Interface.

We have a mix of skills and experience in the team which provides a good dynamic. Those with more experience are able to guide the project, and are able to resolve some issues instinctively. Whilst those with less experience bring new ideas and question current practices, making sure the software is as good as it can be.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOCPL01

DEVELOPMENT PRACTICES

In order to avoid lost beam time, and to provide the instrument scientists with confidence in the system we use a number of development practices to support the production of IBEX.

The development project for IBEX is run based on Agile [12], with a modified SCRUM [13] implementation. The main modification we have undertaken is in relation to the role of the Product Owner. Instead of having a single Product Owner we use the more experienced members of the development team as proxy product owners. This is because each beamline has its own individual requirements and priorities. The team translate these requirements and priorities from the beamline scientists into a smaller prioritised list that can be considered each sprint.

A Scientific Advisory Group (SAG) has been set up with various representatives on it to provide strategic guidance as to the importance of features and development work. Members of the development team have also taken on roles where they will champion an area of work, such as the GUI, or the migration of an instrument, ensuring prioritisation of those tasks, and management of more individual users. Similarly, we do not have a named scrum master, but different people take on the role as required, or based on their skills and knowledge.

Due to these modifications our SCRUM ceremonies, the standard meetings of SCRUM, are also slightly altered. We have a daily stand up that considers the operations element of our role as well as the development areas, looking early for potential issues with our systems. Our sprints run over 4 weeks, but our releases don't coincide with a sprint end. This isn't in keeping with true SCRUM, but is practical for the rhythm of the facility which runs for scheduled experiment cycles. These cycles can have variable times between them, and run for longer than a sprint, and so we aim to release before most cycles. We do not have any 'customers' present at our sprint reviews, but instead the members of the development team take on that role based on knowledge and experience. The changes discussed are later communicated to the scientists in smaller meetings, a few beamlines at a time, by a subset of the development team.

In order to ensure that any code changes are of an appropriate standard, we undertake Continuous Integration (CI) via Jenkins [14], which runs both system and unit tests and creates a deployment package. The builds have been hooked into our git [15] version control system, meaning tests are run before code changes are merged, following best practice for software development.

Part of the system testing is a suite of IOC tests. These ensure that the IOC is can still communicate directly with the hardware device. It is not possible or desirable to replicate all the devices on a beamline so instead we use a device emulator written in LeWIS [16] python framework to emulate devices. The use of emulators allows us to set emulated conditions in the hardware from the mundane, e.g. a temperature value, to the extreme, e.g. an overheating bearing in a chopper. This approach allows for easy regression testing even when dependencies are up-graded, and improves confidence in the code base being deployed initially and on upgrade. Furthermore this technique allows us to undertake Test Driven Development (TDD) of the IOC because the device does not need to be setup every time a set of tests are run. The final stage of development is to check the software against the item being controlled.

We also undertake system testing using Squish [17]. This allows us to look at more integrated testing, and checks that our GUI is still functioning as expected.

The unit and system testing is undertaken frequently thanks to the CI, but there are some system tests we haven't automated yet, and some which have to be run manually. As such, given the size of our code base prior to a release we do undertake a round of system testing. This testing is time consuming, and the slower release schedule compared to our four week sprints makes this task less onerous.

RECENT ADDITIONS

Live View

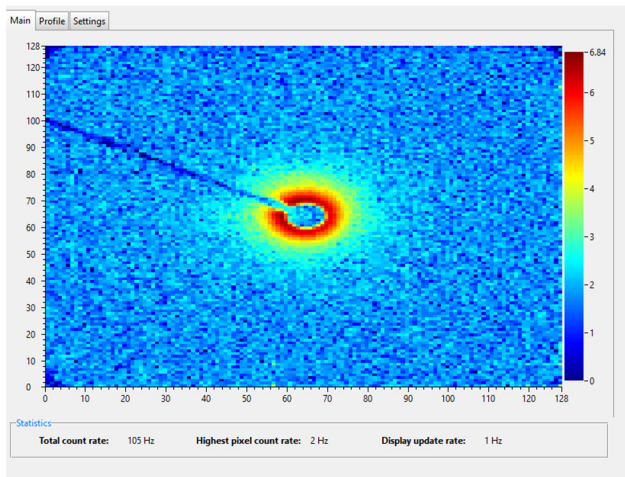

Figure 4: Example of live view image.

There are times when an instrument scientist wishes to see what the neutron data is providing in close to real time, this is what live view provides. The image in Fig. 4 shows data from one of the instruments in the Small Angle Neutron Scattering (SANS) technique suite of instruments. This data is plotted whilst the system is still collecting data, rather than needing to wait for the experiment to finish and then plotting the data, or having to write a file so that the analysis software can then open that interim or experiment file.

Datastreaming

The datastreaming project was started as part of ISIS's in-kind contribution to the ESS. The current neutron data collection model at ISIS is to store data in bespoke hardware, read it out into memory on the instrument server and then collate it with sample environment data and save it to file. This model is not feasible at the ESS due to the very high data rates expected.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOCPL01

Instead the ESS will use a system where individual neutron events are published straight from detectors into a distributed streaming platform based on Kafka [18]. Sample environment data from EPICS will also be published into this Kafka instance. Finally, a separate process will read all the scientific data out of Kafka and collate it into a stored file. As well as being able to deal with greater data rates the system has a number of other advantages including:

- Server redundancy in the Kafka cluster
- Mechanisms to deal with data loss in transit
- More modular design
- Ability to "playback" past neutron events

We are currently in the process of planning how we can roll this system out across ISIS. For more information on the system see [19].

IN PROGRESS

Motion Control

Motion control at ISIS is currently mostly provided by a Galil [20] based system. Although we have found these controllers to be reliable they are becoming unable to meet the accelerating requirements of scientists. This is most noticeable in axis synchronisation, where scientists wish to be able to move a large number of axes (>10) in a synchronised way, whilst maintaining accurate timestamps to correlate motor positions with neutron data. Attempted solutions to this using the Galil required software running on the control VM and the delay of communication with the controller meant that meeting the required accuracy was difficult.

To better meet these requirements we are moving to an Ethercat based system, specifically that provided by Beckhoff twincat [21]. This system contains a PLC that can be used to coordinate motion for a large number of axes in real time as well as gather high precision timestamps. This additional functionality comes with additional complexity, specifically there is a challenge in creating a framework for the PLC code that is flexible enough to be used in a wide variety of applications without requiring bespoke code on every controller. To tackle this complexity we have started working much more closely with mechatronics engineers within ISIS as well as starting a working group with colleagues at the European Spallation Source (ESS) [22] and the Julich Centre for Neutron Science (JCNS) [23].

Reflectometry

One of the more complicated scientific techniques from the beamline controls perspective is reflectometry. This technique measures neutrons reflected off a sample with an extremely flat surface, often a thin film on a substrate. Based on the incident angle of the neutrons, then the components of beamline need to be aligned to ensure that the neutrons are detected accurately. This co-ordinated and complex motion control has required the development of a reflectometry server which calculates the beam path and then derives the required motion for the beamline components.

Server Architecture

It has already been stated that under the existing architecture IBEX is run in a VM on a dedicated server. A VM can be described as a "computer file … that behaves like an actual computer" [24]. The use of this file allows for easier backup and restoration of the system as the host system ceases to be as important to the overall system. This use of a VM was in place at ISIS Pulsed Neutron and Muon Source before we started introducing IBEX. The VM is part of a wider variety of options for virtualisation [25], and with the increase of cloud computing and similar the availability and usability options of virtualisation techniques has increased.

One technique that is being developed at ISIS is to use a collection of Virtual Hard Disks (VHDs) to help with deployment. This allows different part of the system, stored on different disks, to be updated independently, see Fig. 5.

The system VHD will contain the OS and installed programs and applications. The VHD would be updated as required based on the requirements for cybersecurity updates.

The App VHD would house our applications, this is where IBEX is installed, and will ideally be generic to all instruments. Practically IBEX may require hotfixes on individual instruments. These hotfixes are updates that occur when new devices are being developed, or something is changed for an instrument between releases of IBEX. These hotfixes are recoded and then dealt with appropriately for the next release, which might be a merge into master, or a reworking of something to provide more flexibility.

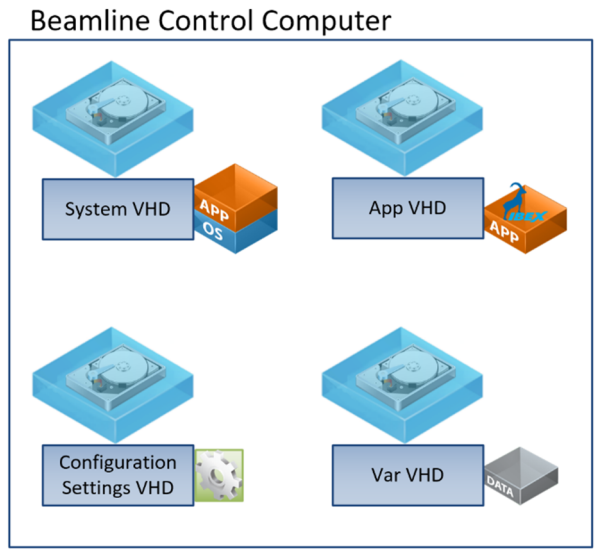

Figure 5: Collection of VHDs.

The Configuration Settings VHD is where the settings for that beamline are kept, such as addresses for sample environment equipment, and will be persisted through upgrades and updates of the System and App VHDs. During updates of IBEX, some items in the settings may also need to be updated, but these changes will be small and will not involve replacing the system. Backing up this VHD should

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOCPL01

allow a new build to run as if it were an existing system, or a system that failed to be easily replaced.

The Var VHD contains the variable data, for example the information generated during experiments. This should be the most volatile of the VHDs, with the contents changing regularly, and important information being archived for access and analysis at a later date.

FUTURE PLANS

Scripting Enhancements

When the ISIS Synchrotron is running, the beam is available for experiments 24 hours every day of the week. The use of scripting allows the useful time to be maximised by running scripts overnight. genie_python allows for long running scripts to be created and run from a terminal, but it is unable to recover should the process be interrupted or the script have an error in it. To avoid this the plan is to queue a number of smaller user scripts onto a script server. If one of these scripts is interrupted, the next one in the queue can be enacted when the script server starts up. Rather than create our own script queueing system we have integrated NICOS [26] in order to use their script server functionality. We have had to create our own user interfaces for it, but the core server functionality is on beamlines and provides a good basis for our script server.

The instrument scientists have asked for some extensions, such as time estimation for how long the script should take to run, and being able to skip the current script. They would also like to see our interfaces include such things as syntax highlighting and code completion. There are also occasions where there is a need to run a secondary script in addition to the primary one. For example, the primary script may be ramping temperatures for experiment data, whilst a secondary script ensures that a motor keeps a stirrer going continuously – we do not want to interrupt the stirrer for the sake of changing a temperature.

Alongside the Script Server is the Script Generator. When an experiment requires changing a few standard settings then being able to automate the generation of the script simplifies the process and reduces errors from typing. Repeated items can be managed without the need for the instrument scientist to enter each step value for a loop, or for a visiting scientist to be familiar with commands in genie_python that undertake data collection. Different techniques have different requirements for the interface and output of the script generator. The generated script also needs to interact easily and transparently with the script server, so that the generator can be given appropriate parameters, create the script and send it to the server to run with the user not looking at any screen other than the generator.

User Interfaces

Whilst it is still very early for us, we are starting to consider the changes that will be required as CS Studio Phoebus replaces CS Studio Eclipse.

There is also a requirement for some beamlines for localised control of equipment, especially motion for beamlines, and we are looking into building clients that will run easily and reliably on tablet computers. The biggest consideration for this usability, allowing for the fact that fingers are less precise than mouse pointers and keyboards.

EPICS 7

The introduction of the structure type for pvData is a highly anticipated addition to our system, and pvAccess will improve some of the interactions between our server and clients. These will both become available as we upgrade our EPICS base software to EPICS 7.

Instrument Migration

We are still migrating instruments to the new system, as well as developing the functionalities previously mentioned. This means we are still adding support for sample environment equipment under IBEX.

CONCLUSIONS

IBEX is well on the way to replacing our old control system, and we hope to complete the process in the next 1-2 years. The change in control system was originally driven by new and more complex experiment requirements, but as part of the process we have updated our development practices too. We are now employing test driven development using device emulators, as well as code reviews and unit testing. Migrating from the previous control system also highlighted issues with its documentation, which we are addressing as part of the IBEX migration.

In any project it is also important to keep customers on board, especially when requirements have to be prioritised. The Scientific Advisory Group and regularly demonstrating new features to individual science groups has helped here, as well as developing and running regular training courses.

Additional benefits from the distributed nature of the new control system are also now also being realised. We are able to put in place cross instrument views and monitoring of key equipment parameters for technical group e.g. detector HV or cryogenic values. In some cases we also place these values into Nagios to generate alerts. We have been able to take advantage of existing EPICS support for imaging cameras, as well as general community written drivers. The reflectometry server will also be more flexible and powerful than the previous implementations.

We have also been looking to improve the way we manage the instrument as a whole, taking advantage of tools to build entire system images and so remove the need to apply security patches or updates to every instrument individually. This will improve reliability and maintainability.

Though scripting is a very powerful tool, writing scripts can be error prone. The script generator in conjunction with the script server will help this. However, as we found from consulting scientists, there are several different user workflows that we will need to provide.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOCPL01

We are yet to take advantage of the new pvAccess protocol available in EPICS 7, but we will be updating our EPICS base software to make such features available very shortly. We will also be deploying additional Kafka based datastreaming instances to allow more live data visualisation and analysis.

REFERENCES

[1] The ISIS Pulsed Neutron and Muon Source, http://www.isis.stfc.ac.uk/

[2] F. A. Akeroyd, *et al*., "IBEX – an EPICS based control system for the ISIS pulsed neutron and muon source," in *Proc. 22nd meeting of the International Collaboration on Advanced Neutron Sources (ICANS XXII)*, Oxford, UK, Mar. 2017, p. 012019.

[3] EPICS, https://epics-controls.org/

[4] Control Systems Studio, http://controlsystemstudio.org/

[5] F.A. Akeroyd *et al.*, "IBEX - the New EPICS Based Instrument Control System at the ISIS Pulsed Neutron and Muon Source", in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, October 2015, paper MOPGF048, pp. 205-208, doi:10.18429/JACoW-ICALEPCS2015-MOPGF048

[6] procServ Process Server, https://sourceforge.net/projects/procserv/.

[7] NeXus Format, https://www.nexusformat.org/.

[8] PCASpy: Portable Channel Access Server in Python, https://pypi.python.org/pypi/pcaspy/.

[9] Genie Python, https://github.com/ISISComputingGroup/genie_python

[10] Open Genie, http://www.opengenie.org/.

[11] DevOps, https://devops.com/definition-devops-masses/.

[12] Agile Manifesto, http://agilemanifesto.org/.

[13] SCRUM, https://www.scrum.org/.

[14] Jenkins, https://jenkins.io/.

[15] Git, https://git-scm.com/.

[16] LeWIS stateful device simulator, http://lewis.readthedocs.io/en/latest/.

[17] Squish, https://www.froglogic.com/squish/.

[18] Kafka, https://kafka.apache.org/.

[19] M. J. Clarke *et al.*, "Live Visualisation of Experiment Data at ISIS and the ESS", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 431-434. doi:10.18429/JACoW-ICALEPCS2017-TUPHA029

[20] Galil, http://www.galilmc.com/.

[21] Beckhoff, https://www.beckhoff.co.uk/.

[22] ESS, https://europeanspallationsource.se/.

[23] JCNS https://www.fz-juelich.de/jcns/.

[24] Virtual Machine, https://azure.microsoft.com/en-gb/overview/what-is-a-virtual-machine/.

[25] Virtualisation, https://azure.microsoft.com/en-gb/overview/what-is-virtualization/.

[26] NICOS, https://forge.frm2.tum.de/nicos/.